

Functions

- A block of code which we can use many times

```
In [12]: num1=10
num2=20
add=num1+num2
print(add)
```

30

```
In [13]: import random
random.randint
```

Out[13]: <bound method Random.randint of <random.Random object at 0x0000025AA46200A0>>

```
In [ ]: def <function_name>():
        codelines
```

Functions with out arguments

```
In [14]: def addition():
num1=10
num2=20
add=num1+num2
print(add)
```

```
In [ ]: # function name: addition
# in order to execute code lines
# we need to call the function
```

```
In [16]: addition()
```

30

```
In [20]: def addition():
num1=10
print("num1 is:",num1)
num2=20
print("num2 is:",num2)
add=num1+num2
print(f"the addition of {num1} and {num2} is: {add}")

addition()
```

num1 is: 10
num2 is: 20
the addition of 10 and 20 is: 30

Note

- function names can be anything

- same rules applicable as variable rules
- whenever you create the function make sure the indentation correctly
- starting with keyword , ending with colon then code lines start with indentation
- brackets means functions
- Never missed the brackets whenever you call the function
- If you want to execute the code we need to call the function
- while you are calling the function if you see function or bound method
- which means you missed the brackets

```
In [ ]: def addition():
        num1=10
        print("num1 is:",num1)
        num2=20
        print("num2 is:",num2)
        add=num1+num2
        print(f"the addition of {num1} and {num2} is: {add}")

        addition()
```

```
In [21]: num1=10
        num2=20
        add=num111+num2
        print(add)
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[21], line 3
      1 num1=10
      2 num2=20
----> 3 add=num111+num2
      4 print(add)

NameError: name 'num111' is not defined
```

```
In [22]: def addition1():
        num1=10
        print("num1 is:",num1)
        num2=20
        print("num2 is:",num2)
        add=num111+num2
        print(f"the addition of {num1} and {num2} is: {add}")
```

```
In [23]: addition1()

num1 is: 10
num2 is: 20
```

```

-----
NameError                                Traceback (most recent call last)
Cell In[23], line 1
----> 1 addition1()

Cell In[22], line 6, in addition1()
      4 num2=20
      5 print("num2 is:",num2)
----> 6 add=num111+num2
      7 print(f"the addition of {num1} and {num2} is: {add}")

NameError: name 'num111' is not defined

```

whenever we defined the function It does not throw any error until unless we call the function

```

In [ ]: # wap ask the user enter 3 numbers calculate average
#-----

# wap ask the user enter radius values find the area of the circle
#-----

# wap ask the user bill amount,
#     ask the user how much tip you want pay in percentage
#     calculate totalbill
#-----

#wap ask the user enter base height calculate area of the triangle
#-----

#wap ask the user enter length and breadth calculate area of the rectangle
#-----

## wap ask the user enter a number
# find it is a even number or odd number
#-----

## wap ask the user enter the distance
# if distance greater than 25km
#     then enter the charge
#     print the total cost
#otherwise
#     print free ride

```

```

In [26]: # wap ask the user enter 3 numbers calculate average
num1=eval(input("enter the num1:"))
num2=eval(input("enter the num2:"))
num3=eval(input("enter the num3:"))
avg=(num1+num2+num3)/3
avg1=round(avg,2)
print(f"the average of {num1} ,{num2} and {num3} is {avg1}")

```

```

enter the num1: 20
enter the num2: 30
enter the num3: 40
the average of 20 ,30 and 40 is 30.0

```

```
In [28]: # function name and variable name should not be same
def average():
    num1=eval(input("enter the num1:"))
    num2=eval(input("enter the num3:"))
    num3=eval(input("enter the num3:"))
    avg=(num1+num2+num3)/3
    avg1=round(avg,2)
    print(f"the average of {num1} ,{num2} and {num3} is {avg1}")

average()
```

```
enter the num1: 20
enter the num3: 30
enter the num3: 40
the average of 20 ,30 and 40 is 30.0
```

- while we are defining function if nothing mentioned means
- It is called as Functions with out arguments

```
In [ ]: def addition()
def average()
def area_of_traingle()
def bill()
```

Functions with arguments

- First look at how many arguments or variables are there
- In that how many input variables are there
- How many output variables are there
- Input variables means user can defined
- output variable means python gives the output
- We can use only input variables as arguments inside the function

```
In [ ]: def addition():
    num1=10
    num2=20
    add=num1+num2
    print(f"the addition of {num1} and {num2} is: {add}")

addition()
```

```
In [29]: def addition2(num1,num2):
    add=num1+num2
    print(f"the addition of {num1} and {num2} is: {add}")

addition2(10,20)
```

the addition of 10 and 20 is: 30

```
In [33]: # try for average

def average1(num1,num2,num3):
    avg=(num1+num2+num3)/3
    avg1=round(avg,2)
    print(f"the average of {num1} ,{num2} and {num3} is {avg1}")

average1(30,40,50)
```

the average of 30 ,40 and 50 is 40.0

```
In [ ]: average1(num1,num2,num3) # Mistake-1
        average1()               # Mistake-2
```

```
In [35]: #wap ask the user enter base height calculate area of the traingle
def average1(num1,num2,num3):
    avg=(num1+num2+num3)/3
    avg1=round(avg,2)
    print(f"the average of {num1} ,{num2} and {num3} is {avg1}")

average1(30,40,50)
```

the average of 30 ,40 and 50 is 40.0

```
In [36]: def area(len,bre):
        a=(len*bre)
        print(f"area is {a}")

        area(10,20)
```

area is 200

```
In [3]: def sub():
        n1=eval(input("enter number1:"))
        n2=eval(input("enter number2:"))
        sub=n1-n2
        print(sub)
        sub()

        # Do not provide function name and variable name both are same
```

enter number1: 12
enter number2: 13
-1

```
In [4]: def sub(n1,n2):
        sub=n1-n2
        print(sub)
        sub(10,20)
```

-10

```
In [ ]: num=eval(input("enter the number:"))
        if num%2==0:
            print(f"the {num} is even")
```

```

else:
    print(f"the {num} is odd")

```

```

In [5]: def even_odd():
        num=eval(input("enter the number:"))
        if num%2==0:
            print(f"the {num} is even")
        else:
            print(f"the {num} is odd")

        even_odd()

```

enter the number: 20
the 20 is even

Default arguments

```

In [6]: # Bill amount problem
        bill=eval(input("enter the bill:"))
        tip_per=eval(input("enter the tip_percentage:"))
        tip_amount=bill*tip_per/100
        total_bill=bill+tip_amount
        print("The total bill is:",total_bill)

```

enter the bill: 1000
enter the tip_percentage: 10
The total bill is: 1100.0

```

In [7]: # with out arguments
        def bill_pay():
            bill=eval(input("enter the bill:"))
            tip_per=eval(input("enter the tip_percentage:"))
            tip_amount=bill*tip_per/100
            total_bill=bill+tip_amount
            print("The total bill is:",total_bill)

        bill_pay()

```

enter the bill: 1000
enter the tip_percentage: 20
The total bill is: 1200.0

```

In [8]: # with arguments
        # whenever you provide the arguments
        # these provided arguments are we using inside the code or not
        def bill_pay(bill,tip_per):
            tip_amount=bill*tip_per/100
            total_bill=bill+tip_amount
            print("The total bill is:",total_bill)

        bill_pay(2000,20)

```

The total bill is: 2400.0

```

In [ ]: # Bill amount problem
        bill=eval(input("enter the bill:"))
        tip_per=eval(input("enter the tip_percentage:"))

```

```

tip_amount=bill*tip_per/100
total_bill=bill+tip_amount
print("The total bill is:",total_bill)

#####
# with out arguments
def bill_pay():
    bill=eval(input("enter the bill:"))
    tip_per=eval(input("enter the tip_percentage:"))
    tip_amount=bill*tip_per/100
    total_bill=bill+tip_amount
    print("The total bill is:",total_bill)

bill_pay()
#####
# with arguments
# whenever you provide the arguments
# these provided arguments are we using inside the code or not
def bill_pay(bill,tip_per):
    tip_amount=bill*tip_per/100
    total_bill=bill+tip_amount
    print("The total bill is:",total_bill)

bill_pay(2000,20)

```

```

In [9]: def bill_pay(bill,tip_per=20):
        print("bill is:",bill)
        print("tip_per is:",tip_per)
        tip_amount=bill*tip_per/100
        total_bill=bill+tip_amount
        print("The total bill is:",total_bill)

        bill_pay(2000)

        # Here the tip_per=default value
        # whenever we provided default value to a arguments
        # Then it is called default argument

```

bill is: 2000
tip_per is: 20
The total bill is: 2400.0

```

In [11]: def average1(num1,num2,num3=50):
        print("num1:",num1) # 30
        print("num2:",num2) # 40
        print("num3:",num3) # 50
        avg=(num1+num2+num3)/3
        avg1=round(avg,2)
        print(f"the average of {num1} ,{num2} and {num3} is {avg1}")

        average1(30,40)

        # num1 num2 are postitional arguments
        # num3 is default argument

```

```
num1: 30
num2: 40
num3: 50
the average of 30 ,40 and 50 is 40.0
```

```
In [12]: def average1(num1,num2=40,num3):
          print("num1:",num1) # 30
          print("num2:",num2) # 40
          print("num3:",num3) # 50
          avg=(num1+num2+num3)/3
          avg1=round(avg,2)
          print(f"the average of {num1} ,{num2} and {num3} is {avg1}")

          average1(30,50)
```

Cell In[12], line 1

```
def average1(num1,num2=40,num3):
```

^

SyntaxError: non-default argument follows default argument

```
In [13]: def average1(num1,num3,num2=40):
          print("num1:",num1) # 30
          print("num2:",num2) # 40
          print("num3:",num3) # 50
          avg=(num1+num2+num3)/3
          avg1=round(avg,2)
          print(f"the average of {num1} ,{num2} and {num3} is {avg1}")

          average1(30,50)
```

```
num1: 30
num2: 40
num3: 50
the average of 30 ,40 and 50 is 40.0
```

Note

- Default argument always at last

```
In [ ]: average1(num1,num2,num3=40) # W
         average1(num1,num2=50,num3) # F
         average1(num1=100,num2,num3) # F
         average1(num1,num2=50,num3=40) # W
         average1(num1=100,num2,num3=40) # F
         average1(num1=100,num2=50,num3) # F
         average1(num1=100,num2=50,num3=40) # W
```

```
In [14]: def average1(num1,num2=50,num3=40):
          print("num1:",num1) # 30
          print("num2:",num2) # 40
          print("num3:",num3) # 50
          avg=(num1+num2+num3)/3
          avg1=round(avg,2)
          print(f"the average of {num1} ,{num2} and {num3} is {avg1}")
```



```
average1(30)
```

```
num1: 30
num2: 50
num3: 40
the average of 30 ,50 and 40 is 40.0
```

```
In [15]: # Case-1:
def addition(n1,n2=600):
    add=n1+n2
    print(add)

addition(500)
```

```
1100
```

```
In [16]: # Case-2:
def addition(n1,n2=600):
    add=n1+n2
    print(add)

addition(500,1000)

# First we are defining function
# while define the function we given n2=600
# now we are calling the function,
# while we are calling n2=1000
# So value will be overwrite
# Python always takes the latest value
```

```
1500
```

```
In [17]: # Case-3:
def addition(n1,n2=600):
    n2=2000
    add=n1+n2
    print(add)

addition(500,1000)

#A) 1100 B) 1500 C) 2500 D) error

# Step-1: Define the function n2=600
# Step-2: Call the fuction n2=1000
# Step-3: Running the function n2=2000
```

```
2500
```

```
In [ ]: # Case-4:
def addition(n1,n2=600):
    n2=2000
    add=n1+n2
    print(add)
n2=3000
addition(500,1000)
# n2=600 ==== > 3000 === >1000m==== > 2500
```

Local variable and Global variable

- Local variable means : the variables inside the function call
- Global variable means: the variables outside the function call
- Once you define the variables outside means , you can use those variables anywhere
- You can use global variables inside function also
- But you can not use local variables outside the function

```
In [18]: def multiplication():  
         a=10  
         b=20  
         mul=a*b  
         print(mul)  
         multiplication()  
  
         # a,b,mul are Local variables  
         # these values we can not use outside the function
```

200

```
In [20]: mul
```

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[20], line 1  
----> 1 mul  
  
NameError: name 'mul' is not defined
```

```
In [24]: a1=100  
         b1=200  
         def multiplication1():  
             mul1=a1*b1  
             print(mul1)  
             multiplication1()
```

20000

```
In [22]: a1
```

```
Out[22]: 100
```

```
In [25]: mul1
```

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[25], line 1  
----> 1 mul1  
  
NameError: name 'mul1' is not defined
```

```
In [26]: # Case-5:
a1=100
b1=200
def add11(a1=1000):
    add1=a1+b1
    print(add1)
add11()

# a1=100 == > a1=1000 == > 1000+200

#Step-1: Global variable intialization a1=100
# Step-2: Define the function a1=1000
# Step-3: Call the function a1=1000
# Step-4: Running the function a1=1000
```

1200

```
In [27]: # Case-6:
a1=100
b1=200
def add11(a1=1000):
    add1=a1+b1
    print(add1)
add11(a1=2000,b1=3000)

# while defiening function only one argument is there
# while calling we are providing two arguments
# It is not possible
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[27], line 7
      5     add1=a1+b1
      6     print(add1)
----> 7 add11(a1=2000,b1=3000)

TypeError: add11() got an unexpected keyword argument 'b1'
```

```
In [28]: # Case-7:
a1=100
b1=200
def add11(a1=1000):
    add1=a1+b1
    print(add1)
add11(a1=2000)
```

2200

```
In [31]: # Case-8:
a1=100
b1=200
def add11(a1=1000):
    a1=5000
    add1=a1+b1
    print(add1)
add11(a1=2000)
```

5200

In [32]: a1

Out[32]: 100

```

In [33]: # Case-9:
a1=100
b1=200
def add11(a1=1000):
    a1=5000
    add1=a1+b1
    print(add1)
a1=6000
add11(a1=2000)
# a1=100,b1=200
# define the function a1=1000
# a1=6000
# calling the a1=2000
# running a1=5000

```

5200

In [34]: a1

Out[34]: 6000

```

In [35]: # Case-10:
b1=200
def add11():
    add1=a1+b1
    print(add1)
a1=6000
add11()

# b1=200
# define the function
# a1=6000
# calling the function
# running the function

```

6200

```

In [ ]: # Case-11:
b1=200
def add11():
    add1=a1+b1
    print(add1)
a1=6000
add11()

```

In []: even through we use same variable name a1 in local and global.. both are different

```

In [36]: a=100
def greet():
    a=200

```

```
print('hello')
greet()
```

hello

```
In [ ]: # step-1: Variables intilised
        # Step-2: Define the function
        # Step-3: Calling the function
        # Step-4: Running the function
```

```
In [38]: # Case-12:
        b21=200
        def add21():
            add1=a21+b21
            print(add1)
        add21()
        a=6000
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[38], line 6
      4     add1=a21+b21
      5     print(add1)
----> 6 add21()
      7 a=6000

Cell In[38], line 4, in add21()
      3 def add21():
----> 4     add1=a21+b21
      5     print(add1)

NameError: name 'a21' is not defined
```

```
In [1]: s=0
        def add():
            s=s+10
            print(s)
        add()
```

```
-----
UnboundLocalError                        Traceback (most recent call last)
Cell In[1], line 5
      3     s=s+10
      4     print(s)
----> 5 add()

Cell In[1], line 3, in add()
      2 def add():
----> 3     s=s+10
      4     print(s)

UnboundLocalError: cannot access local variable 's' where it is not associated with
a value
```

```
In [2]: s=0
        def add():
```

```

a=s+10
print(a)
add()

```

10

```

In [3]: b=0
def add():
    s=b+10
    print(s)
add()

```

10

```

In [4]: s=0 # GV
def add():
    c=s+10
    s=c # LV
    print(s)
add()

```

UnboundLocalError

Traceback (most recent call last)

```

Cell In[4], line 6
      4 s=c # LV
      5 print(s)
----> 6 add()

```

```

Cell In[4], line 3, in add()
      2 def add():
----> 3 c=s+10
      4 s=c # LV
      5 print(s)

```

UnboundLocalError: cannot access local variable 's' where it is not associated with a value

```

In [5]: n1=10
def addition():
    n1=100
    n2=200
    n3=n1+n2
    print(n1,n2,n3)

addition()

#

```

100 200 300

Note

- if you give same variabel as global and local
- make sure that that variable should not use as value inside the function

```
In [7]: n11=10
def addition():
    n22=200+n11
    n33=n11+n22
    print(n11,n22,n33)

addition()

# step-1: n11 =10
# step-2: define the function
# step-3: call the function
# step-4: run the function
#         n11 is global variable passing inside the function (yes)
#         n22=200+n11= 200+10=210
#         n33=10+210=220
#         print(10,210,220)
```

10 210 220

```
In [ ]: s=0 # GV
def add():
    c=s+10
    s=c # LV
    print(s)
add()

# step-1: s=0    gv
# step-2: define the function
# step-3: call the function
# step-4: run the function
#         c is lv : c= s+10 = c=10
#         we are creating a new variable same like as global s
# we are creating a new variable with same name as global variable
# by using global variable
```

```
In [9]: value=100
def greet():
    value=200
    print('hello')
greet()
```

hello

```
In [10]: value
```

```
Out[10]: 100
```

```
In [ ]: **global**

- global keyword is used to take the local variable value outside the function
```

```
In [11]: value=100
def greet():
    global value
```

```

    value=200
    print('hello')
    greet()
    print('value:',value)

```

hello
value: 200

```

In [15]: n11=10
         def addition():
             global n33,n22
             n22=200
             n33=n11+n22
             print(n11,n22,n33)

         addition()

```

10 200 210

```

In [16]: n33

```

Out[16]: 210

- Outside the function is called Global variables
- Inside the function is local variable
- local variables can not use outside the function
- if you want to use outside the function use **global** keyword

```

In [18]: def mul(a,b):
         print(a*b)

         mul(eval(input()),
             eval(input()))

```

20
30
600

Return

- we can use the local variable or function outputs outside the function using return
- print is different
- return is different
- print is used to only to print the values
- that value you can only see, but you can not use

- if you want to use outside we will use return (it is also possible by using global)

```
In [21]: def average():
        n1=10
        n2=20
        n3=30
        avg=(n1+n2+n3)/3
        return(avg)
        avg=average()

        # function is ready to return values to me
        # so i need to store
```

```
In [22]: avg
```

```
Out[22]: 20.0
```

```
In [23]: def average():
        n1=10
        n2=20
        n3=30
        avg=(n1+n2+n3)/3
        return(avg,n3)
        avg,n3=average()

        # How many values we are returning
        # we will store each return values in different variable
        # because two values are different
        # If you use only variable to store : tuple value
```

```
In [24]: print(avg)
        print(n3)
```

```
20.0
30
```

```
In [29]: def sub():
        a=10
        b=20
        subb=b-a
        return(subb)

        subb=sub()
```

```
In [28]: subb
```

```
Out[28]: 10
```

```
In [ ]: # Return always at last line of the function
        # Not in middle lines

        #Sir in real time projects, do we use return or global more sir?
        #return
```

```
In [34]: def sub():
          a=10
          b=20
          subb=b-a
          return(a,b,subb)
          out=sub()
```

```
In [35]: out
```

```
Out[35]: (10, 20, 10)
```

```
In [ ]: - with out arguments

        - with arguments

        - default arguments

        - local vs global

        - global keywords

        - return

        - function in function

        - *kwargs: keyword arguments
```

unbound local error

```
In [6]: num=10 # gv
        def fun1():
            num1=100 # lv
            print("inside function:",num1) # print(100)

        fun1()
        print("outside function:",num1)
```

```
inside function: 100
outside function: 10
```

```
In [4]: num2=10# gv
        def fun2():
            print(num2)
            num2=100 # lv

        fun2()
        # Name error wil come
        # Local variable and global variable both names are same
        # you are try to access local variable before assign
```

UnboundLocalError

Traceback (most recent call last)

Cell In[4], line 6

```

3     print(num2)
4     num2=100
----> 6 fun2()

```

Cell In[4], line 3, in fun2()

```

2 def fun2():
----> 3     print(num2)
4     num2=100

```

UnboundLocalError: cannot access local variable 'num2' where it is not associated with a value

```

In [ ]: s=0 # GV
def add():
    c=s+10
    s=c # LV
    print(s)
add()

```

In []: sir but num2=10, its is a global variable no sir
 but on Mondays class it worked no sir
 first initialization happens, define the function
 call the function
 so sir num2=20 and it is not updated

```

In [ ]: a1=100
b1=200
def add11():
    add1=a1+b1
    print(add1)
add11()

```

```

In [9]: num2=10# gv
def fun2():
    global num2
    num2=100 # Lv

fun2()

```

```

In [10]: num2

```

```

Out[10]: 100

```

```

In [14]: a1=100
b1=200
def add11():
    add1=a1+b1
    print(add1)
add11()

# Here No local variables as same name as global variable

```

```
#####
a1=100
b1=200
def add12():
    a1=700
    b1=800
    add1=a1+b1
    print(add1)
add11()

# we are intialisng lv names same as gv
# we are using lv after intialization
#####
a1=100
b1=200
def add13():
    add1=a1+b1
    a1=700
    b1=80
    print(add1)
add11()
# we are intialisng lv names same as gv
# we are using lv before intialization
```

300

No error

- Here No local variables as same name as golbal variable
- we are intialisng lv names same as gv
 - we are using lv after intialization

Error

- we are intialisng lv names same as gv
 - we are using lv before intialization

```
In [17]: a1=100
b1=200
def add12():
    c=700
    d=800
    add1=c+d
    print(add1)
add12()
```

```
a1=100
b1=200
def add32():
    add1=c+d
```

```

c=700
d=800
print(add1)
add32()

```

1500

UnboundLocalError

Traceback (most recent call last)

Cell In[17], line 18

```

16     d=800
17     print(add1)
--> 18 add32()

```

Cell In[17], line 14, in add32()

```

13 def add32():
--> 14     add1=c+d
15     c=700
16     d=800

```

UnboundLocalError: cannot access local variable 'c' where it is not associated with a value

```

In [22]: # Case-12:
b21=200
def add21():
    a21=1000
    add1=a21+b21
    b21=2000
    print(add1)
add21()

```

UnboundLocalError

Traceback (most recent call last)

Cell In[22], line 8

```

6     b21=2000
7     print(add1)
--> 8 add21()

```

Cell In[22], line 5, in add21()

```

3 def add21():
4     a21=1000
--> 5     add1=a21+b21
6     b21=2000
7     print(add1)

```

UnboundLocalError: cannot access local variable 'b21' where it is not associated with a value

- Global variable
- define the function
- call the function
- run the function

Functions in Functions

```
In [23]: def greet1():
          print('hello good morning')

          def greet2():
              print('Good night!')

          greet1()
          greet2()
```

hello good morning
Good night!

```
In [24]: def greet2():
          print('Good night!')

          def greet1():
              print('hello good morning')
              greet2()

          greet1()
          # hello gm
          # good night
```

hello good morning
Good night!

```
In [ ]: def greet2():
          print('Good night!')
          greet1()

          def greet1():
              print('hello good morning')
              greet2()

          greet1()

          # Hello gm
          # greet2 ==== GN
          # greet1 ==== hello gm
```

- Functions with out arguments
- Functions with arguments
- Functions default arguments
- Global variable vs Local variable
- Global keyword
- return statement

- unbound local error
- Function in Functions

```
In [ ]: distance=eval(input("enter the distance in km:"))
cutoff_distance=eval(input("enter the cuto ff distance in km:"))
if distance>cutoff_distance:
    chargeble_distance=distance-cutoff_distance
    print("kudos to you the chargeble distance is:",chargeble_distance)
    charge=eval(input("enter the charge in rs"))
    cost=chargeble_distance*charge
    print("the total charge is:",cost)
else:
    print("enjoy the free ride")

# 50K upto 25km is free ride
# 50-25=25km
```

```
In [26]: def total_fare():
    distance=eval(input("enter the distance in km:"))
    cutoff_distance=eval(input("enter the cuto ff distance in km:"))
    if distance>cutoff_distance:
        chargeble_distance=distance-cutoff_distance
        print("kudos to you the chargeble distance is:",chargeble_distance)
        charge=eval(input("enter the charge in rs"))
        cost=chargeble_distance*charge
        print("the total charge is:",cost)
    else:
        print("enjoy the free ride")

total_fare()
```

```
enter the distance in km: 60
enter the cuto ff distance in km: 25
kudos to you the chargeble distance is: 35
enter the charge in rs 2
the total charge is: 70
```

```
In [27]: def total_fare1(distance,cutoff_distance,charge):
    if distance>cutoff_distance:
        chargeble_distance=distance-cutoff_distance
        print("kudos to you the chargeble distance is:",chargeble_distance)
        cost=chargeble_distance*charge
        print("the total charge is:",cost)
    else:
        print("enjoy the free ride")

total_fare1(60,25,3)
```

```
kudos to you the chargeble distance is: 35
the total charge is: 105
```

```
In [29]: def total_fare2(distance,cutoff_distance=25,charge=5):
    if distance>cutoff_distance:
        chargeble_distance=distance-cutoff_distance
```

```

    print("kudos to you the chargeble distance is:",chargeble_distance)
    cost=chargeble_distance*charge
    print("the total charge is:",cost)
else:
    print("enjoy the free ride")

total_fare2(100)

```

kudos to you the chargeble distance is: 75

the total charge is: 375

```

In [30]: distance=eval(input("enter the distance in km:"))
cutoff_distance=eval(input("enter the cuto ff distance in km:"))
def total_fare3():
    if distance>cutoff_distance:
        chargeble_distance=distance-cutoff_distance
        print("kudos to you the chargeble distance is:",chargeble_distance)
        charge=eval(input("enter the charge in rs"))
        cost=chargeble_distance*charge
        print("the total charge is:",cost)
    else:
        print("enjoy the free ride")

total_fare3()

```

enter the distance in km: 150

enter the cuto ff distance in km: 50

kudos to you the chargeble distance is: 100

enter the charge in rs 10

the total charge is: 1000

```

In [31]: distance=eval(input("enter the distance in km:"))
cutoff_distance=eval(input("enter the cuto ff distance in km:"))
def total_fare4():
    global cost
    if distance>cutoff_distance:
        chargeble_distance=distance-cutoff_distance
        print("kudos to you the chargeble distance is:",chargeble_distance)
        charge=eval(input("enter the charge in rs"))
        cost=chargeble_distance*charge
        print("the total charge is:",cost)
    else:
        print("enjoy the free ride")

total_fare4()

print("outside function cost is:",cost)

```

enter the distance in km: 100

enter the cuto ff distance in km: 50

kudos to you the chargeble distance is: 50

enter the charge in rs 5

the total charge is: 250

outside function cost is: 250

```

In [33]: distance=eval(input("enter the distance in km:"))
cutoff_distance=eval(input("enter the cuto ff distance in km:"))

```



```
def total_fare5():
    if distance > cutoff_distance:
        chargeble_distance = distance - cutoff_distance
        print("kudos to you the chargeble distance is:", chargeble_distance)
        charge = eval(input("enter the charge in rs"))
        cost = chargeble_distance * charge
        print("the total charge is:", cost)
    else:
        cost = 100

    return(cost)

cost = total_fare5()

print("outside function cost is:", cost)
```

enter the distance in km: 20
 enter the cutoff distance in km: 25
 outside function cost is: 100

In [35]:

```
distance = eval(input("enter the distance in km:"))
cutoff_distance = eval(input("enter the cutoff distance in km:"))
def total_fare5():
    if distance > cutoff_distance:
        chargeble_distance = distance - cutoff_distance
        print("kudos to you the chargeble distance is:", chargeble_distance)
        charge = eval(input("enter the charge in rs"))
        cost = chargeble_distance * charge
        print("the total charge is:", cost)
        return(cost)
    else:
        cost = 100
        return(cost)

cost = total_fare5()

print("outside function cost is:", cost)
```

enter the distance in km: 10
 enter the cutoff distance in km: 25
 outside function cost is: 100

In []:

```
# First create 4 individual function
# Fun1: add
# Fun2: sub
# Fun3: mul
# Fun4: div

def add(a,b):
    print(a+b)

# Create a main function name : calculator
# Inside main function
# Print some statements
```

```
# option1: addition
# option2: sub, opt3: mul opt4: div
# option=eval(input('1-4'))
# a value
# b value
# if option==1:
#     call add function
# elif option==2:
#     call sub function
# elif option==3:
#     call mul function
# elif option==4:
#     call div function
```

```
In [ ]: def add(a,b):
        print(a+b)

def add(a,b):
    print(a+b)

def add(a,b):
    print(a+b)

def add(a,b):
    print(a+b)

def calcualtor():
    # print
    # option
    # a
    # b
    if op==1:
        add(a,b)
```